

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

EV205822975


Generation and Validation of Short Digital Signatures with Implicit Message Embedding

Inventor(s):

Ramarathnam Venkatesan

Peter L. Montgomery

421 West Riverside, Suite 500
Spokane, WA 99201
P: 509.324-9256
F: 509.323-8979
www.lee&hayes.com

The logo for the law firm Lee & Hayes, featuring the firm's name in a stylized font with an ampersand between the words.

ATTORNEY's DOCKET NO. MS1-1285US

1
2 **GENERATION AND VALIDATION OF SHORT DIGITAL SIGNATURES**
3 **WITH IMPLICIT MESSAGE EMBEDDING**

4
5 **TECHNICAL FIELD**

6 This invention generally relates to a technology for cryptography.

7
8 **BACKGROUND**

9 For as long as information has been communicated between two
10 individuals, it has always been susceptible to third-party interception,
11 eavesdropping, compromise and/or corruption. Traditionally, this problem has
12 been handled through the development, over the years, of increasingly
13 sophisticated cryptographic techniques.

14 One class of these techniques involves the use of key-based ciphers. In
15 particular, through a key-based cipher, sequences of intelligible data (i.e.,
16 “plaintext”) that collectively form a message are each mathematically transformed,
17 through an enciphering algorithm, into seemingly unintelligible data (i.e., so-
18 called “ciphertext”).

19 Such transformations are typically completely reversible. This means that
20 the enciphering algorithm is invertible: each ciphertext can be transformed back to
21 its corresponding original plaintext, and each element of plaintext can be
22 transformed into one and only one element of ciphertext.

23 In addition, it is desirable for a particular cipher that generated any given
24 ciphertext to be sufficiently secure from cryptanalysis. To provide a requisite level
25 of security, typically a unique key is selected which defines a unique

1 corresponding cipher. This precludes, to the extent possible, a situation where
2 multiple differing keys each yields reversible transformations between the same
3 plaintext-ciphertext correspondence.

4 The strength of any cryptographic technique (and hence the degree of
5 protection it affords from third-party intrusion) is directly proportional to the time
6 required, by a third party, to perform cryptanalysis. While no encryption technique
7 is completely impervious from cryptanalysis with unlimited resources, ensuring
8 that without the secret key an immense number of calculations and an extremely
9 long time interval are required with today's computing technology effectively
10 rendering many techniques, for all practical intents and purposes, sufficiently
11 secure to warrant their widespread adoption and use.

12 However, computing technology and cryptanalytic techniques continue to
13 rapidly evolve. Processors, unheard of just a few years ago in terms of their high
14 levels of sophistication and speed, are becoming commercially available at ever
15 decreasing prices. What might have taken years of continual computing a decade
16 ago can now be accomplished in a very small fraction of that time. Hence, as
17 technology evolves, the art of cryptography advances in lockstep in a continual
18 effort to develop increasingly sophisticated cryptographic techniques that
19 withstand correspondingly intensifying cryptanalysis.

20 However, encryption, by itself, provides no guarantee that an enciphered
21 message can not be or has not been compromised during transmission or storage
22 by a third party. Encryption does not assure integrity. An encrypted message could
23 be intercepted and changed, even though it may be, in any instance, practically
24 impossible, to cryptanalyze.
25

1 In that regard, the third party could intercept, or otherwise improperly
2 access, a ciphertext message, then substitute a predefined illicit ciphertext block(s)
3 which that party, or someone else acting in concert with that party, has specifically
4 devised for a corresponding block(s) in the message; and thereafter, transmit that
5 resulting message with the substituted ciphertext block(s) onward to a destination.
6 All of this may be done without the knowledge of the eventual recipient of the
7 message and to the eventual detriment of the original message sender and/or its
8 recipient.

9 For example, if the message involved a financial transaction between a
10 purchaser and a seller, the substituted block could be an enciphered account
11 number of a third party rather than that of the intended seller; hence, with an
12 eventual effect of possibly illicitly diverting money originally destined to the seller
13 to the third party instead. For a variety of reasons, messages carried over the
14 Internet are vulnerable in this regard.

15 Detecting altered communication is not confined to Internet messages. With
16 the burgeoning use of stand-alone personal computers, very often, an individual or
17 business will store confidential or other information within the computer, such as
18 on a hard-disk therein, with a desire to safeguard that information from illicit
19 access and alteration by third-parties.

20 Password controlled access--which is commonly used to restrict access to a
21 given computer and/or a specific file stored thereon--provides a certain, but rather
22 rudimentary, form of file protection. Often users are cavalier about their
23 passwords, either in terms of safeguarding their password from others or simply
24 picking passwords that others can easily discern; thereby creating a security risk.
25 Once password protection is circumvented, a third party can access a stored file

1 and then change it, with the owner of the file then being completely oblivious to
2 any such change.

3 **SUMMARY**

5 Described herein is a technology generally related to cryptography.

6 An implementation of a digital signature technique, described herein,
7 generates, and another implementation of a digital signature technique, also
8 described herein, validates, a hidden plaintext or ciphertext message wherein one
9 or more portions of that message have another ciphertext message implicitly
10 embedded therein. In other implementations, two or more portions of that
11 message have another ciphertext message implicitly embedded therein.

12 This summary itself is not intended to limit the scope of this patent.
13 Moreover, the title of this patent is not intended to limit the scope of this patent.
14 For a better understanding of the present invention, please see the following
15 detailed description and appending claims, taken in conjunction with the
16 accompanying drawings. The scope of the present invention is pointed out in the
17 appending claims.

18 **BRIEF DESCRIPTION OF THE DRAWINGS**

19
20 The same numbers are used throughout the drawings to reference like
21 elements and features.

22 Fig. 1-3 are flow diagrams showing methodological implementations
23 described herein.

24 Fig. 4 is an example of a computing operating environment capable of
25 (wholly or partially) implementing at least one embodiment described herein.

DETAILED DESCRIPTION

In the following description, for purposes of explanation, specific numbers, materials and configurations are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that the present invention may be practiced without the specific exemplary details. In other instances, well-known features are omitted or simplified to clarify the description of the exemplary implementations of present invention, thereby better explain the present invention. Furthermore, for ease of understanding, certain method steps are delineated as separate steps; however, these separately delineated steps should not be construed as necessarily order dependent in their performance.

The following description sets forth one or more exemplary implementations of **Generation and Validation of Short Digital Signatures with Implicit Message Embedding** that incorporate elements recited in the appended claims. These implementations are described with specificity in order to meet statutory written description, enablement, and best-mode requirements. However, the description itself is not intended to limit the scope of this patent.

The inventors intend these exemplary implementations to be examples. The inventors do not intend these exemplary implementations to limit the scope of the claimed present invention. Rather, the inventors have contemplated that the claimed present invention might also be embodied and implemented in other ways, in conjunction with other present or future technologies.

An example of an embodiment of **Generation and Validation of Short Digital Signatures with Implicit Message Embedding** may be referred to as an

1 “exemplary short digital signature generator/validator”. Alternatively, an example
2 embodiment of a generator may be referred to as an “exemplary short digital
3 signature generator”, and an example embodiment of a validator may be referred
4 to as an “exemplary short digital signature validator”.

5 Those who are skilled in the art are directed to find additional useful and
6 relevant information in the following co-owned US Patent No 6,209,093, issued
7 Mar 27,2001, titled “Technique For Producing A Privately Authenticatable Product
8 Copy Indicia And For Authenticating Such An Indicia”.

9 The one or more exemplary implementations, described herein, of the
10 present claimed invention may be implemented (in whole or in part) by a
11 computing environment like that shown in Fig. 4.

12 13 **Product Identification (PID) Code**

14 Since software is so often and so easily reproduced, software manufacturers
15 typically require a validation process to enable full functionality of their product.
16 That validation process typically includes the use of a special code, that, when
17 manually entered, enables the full functionality of the product.

18 This often involves the use of a special ciphertext that is typically imprinted
19 on the media’s case or on the accompanying literature. Commonly, this ciphertext
20 is called a product identification (PID). Since this PID is typically manually
21 entered, it is typically desirable to shorten the length of the PID; thereby,
22 improving the overall customer’s installation experience.

23 One conventional PID validation procedure includes the following:
24
25

- When prompted to do so during the software installation process, the user manually enters the PID found on the media's case found in the retail package of the software;
- The installation software validates the PID entered using secret cryptographic tests.
- Alternatively, the target computer may communicate with a central server computer associated with the software manufacturer. That communication may seek validation from the server based upon the manually entered PID.
- Upon validation, the software's full functionality is enabled.

Introduction

The exemplary short digital signature generator, described herein, generates, and the exemplary short digital signature validator, also described herein, validates, a ciphertext message wherein at least two portions of that message have another ciphertext message implicitly embedded therein.

As used herein, the word implicit refers to the fact the existence of a hidden plaintext or ciphertext whose content is not readily apparent in the ciphertext (an encoding of the signed message) itself. Furthermore, in, at least one embodiment, it refers to such a ciphertext message where knowledge of the signer's private key (of a public-private key infrastructure) is insufficient to discover the implicit message in the ciphertext.

In at least one embodiment, the exemplary short digital signature generator/validator seeks to provide an opportunity to find a balance between maximizing the customer's experience (by minimizing the length of the manually

1 entered ciphertext PID) while minimizing the PID's vulnerability to cryptanalysis
2 and other attacks. In other words, it is desirable for the ciphertext PID to be short
3 enough that the customer will tolerate manually entering it, but long enough
4 and/or complicated enough that the ciphertext PID provides a high degree of
5 security from digital pirates.

6 7 **Exemplary Ciphertext Generation**

8 With at least one embodiment, the exemplary short digital signature
9 generator is a central server that may also be called the "signer". For a software
10 manufacturer, it is the central system to generate each PID associated with a
11 specific manufactured product.

12 The exemplary short digital signature generator produces a digital signature
13 of a given message M encoding a pair $\langle M_1, M_2 \rangle$ as $(M_1, r, s, auth)$ with the M_2 part
14 implicitly embedded in (r, s) .

15 If, for example, $|s| = 2L$ and $|r| = L$, the security of the system may be
16 approximately 2^L operations on the elliptic curve group (rather than hash
17 functions) using some standard assumptions.

18 In one embodiment, a rather low value $L = 33$ is chosen and extra design
19 considerations are presented to make it more difficult for an attacker to optimize
20 her computations.

21 Herein, *auth* is an authenticity tag. This tag adds another level of security
22 to the ciphertext. But the relationship between *auth* and a pair $\langle M_1, M_2 \rangle$ is
23 established by a random (or pseudorandom) function which is private to the
24 signer.
25

1 In the event (which is more likely the smaller L is) of a public key being
2 compromised, the probability that a signature generated by a pirate will get
3 through a server is $2^{-\ell}$ where ℓ is the length of the authenticity tag attached to the
4 signature—this signature can be verified at a central server.

5 In the discussion below, G is a fixed group and g is a fixed element of order
6 q in G . The exemplary short digital signature generator uses a special secret key
7 BK which is a random (or pseudorandom) binary string that is sufficiently long
8 (e.g., 128 bits). The exemplary short digital signature generator also uses four
9 predefined hash functions H_0, H_1, H_2, H_3 that are instantiated by using keyed
10 versions of a secure hash function H (e.g. SHA-1). Thus $H_i(x) = H(\text{Key}_i, x)$.

11 An example pseudo-code for a function for generating an implicit
12 ciphertext message (in accordance with the exemplary short digital signature
13 generator) when given a message M which has already been divided into M_1 and
14 M_2 is provided here:

15
16
17 $SIGN(M_1, M_2)$

18 Find a k with $H_0(M_1, g^k) = M_2$.

19 $r = H_0(M_1, g^k)$

20 $s = k/(r + 1) - x H_2(M_1, g^k) \mod q$

21 $auth = H_3(BK, g^k)$

22 return $(M_1, r, s, auth)$, which is one embodiment of the digital
23 signature
24
25

1 Here x is the signer's secret exponent. Note that here M_1 and M_2 are parts
2 of the input to the above exemplary pseudo-code of a methodological
3 implementation (in accordance with the exemplary short digital signature
4 generator).

5 The above methodological implementation assumes that many or all
6 possible values of the second parameter M_2 will occur within the messages that
7 need be signed by this methodological implementation, while the first parameter
8 M_1 stays fixed. It creates an array large enough to accommodate all possible
9 values of the second parameter M_2 .

10 The task of finding a k with $H_0(M_1, g^k) = M_2$ in the first line can be done
11 efficiently using the so called "coupon collector" principle. That is, starting with a
12 fixed M_1 , one picks a value of k and computes the corresponding hash value $M_2 =$
13 $H_0(M_1, g^k)$. Save k in a table indexed by M_2 . Select more k 's until there is a
14 known k for every possible M_2 . Use this k when signing a message that can be
15 viewed as a concatenation of binary strings denoting M_1 and M_2 . In short, the
16 coupon collector principle says that if the H_0 function behaves randomly and if
17 one tries this hashing step slightly more often than the number of possibilities for
18 M_2 then for every value of M_2 there will be at least one trial value of k that
19 satisfies the equation—with a good probability.

20 Alternative implementations may employ fewer bits for r, s while
21 increasing that group size.
22
23
24
25

Methodological Implementation

Fig. 1 shows a methodological implementation of the exemplary short digital signature generator. This methodological implementation may be performed in software, hardware, or a combination thereof.

At 110 of Fig. 1, the exemplary short digital signature generator obtains a message M having two portions, wherein M_1 is one of the portions of the M and M_2 is another.

At 120-140, the exemplary short digital signature generator generates one or more codes (e.g., (r,s)) having a combination with M_2 implicitly embedded therein. In some implementations, more than one code is used. Examples of codes in this context include digitally signed message, codewords, or ciphertext.

The exemplary short digital signature generator implicitly embeds some portion (here, it is M_2) of the message M into a digital signature (DS). The length of M_2 may be, for example, 20 bits. Otherwise, every message would have $2^{|k|} = 2^{64}$ possible valid signatures. In this example, there are only $2^{64-20} = 2^{44}$ possible values of g^k .

At 120, the exemplary short digital signature generator selects an initial value of a variable per-message key (k).

At 130, it tests to see whether where a predefined mathematical function employing M_1 and g^k (for the selected value of k) produces a value equal to M_2 . If not, then it returns to 120 to select a new value of k . If so, then it goes to block 140.

At blocks 120-130, the exemplary short digital signature generator is finding a value of a variable per-message key (k) where a predefined mathematical

1 function employing M_1 and g^k produces a result equivalent to M_2 . This function
2 may include hashing.

3 At 130, it tests whether the result is equivalent to M_2 . If not, it returns to
4 block 120 and selects a new value of k . The selection of a new value of k may be
5 accomplished using many suitable approaches. For example, it may selected
6 randomly, pseudorandomly, sequentially, using a fixed pattern, within a fixed
7 number field, or along a predefined mathematical formula (or curve).

8 If k is equivalent, then it goes the next block 140. At 140, it calculates the
9 one or more codes (e.g., $(r,s,auth)$), where the calculation of one code is not
10 identical to the calculation of any other code and where each calculation
11 incorporates k . These code calculations do not employ M_2 . Therefore, M_2 cannot
12 be derived from reverse engineering these code calculations. These calculations
13 may include hashing. In some implementations, more than one code is used.
14 Examples of codes in this context include digitally signed message, codewords, or
15 ciphertext.

16 At 150, the exemplary short digital signature generator produces a ciphertext
17 signature for the original message M . It produces a digital signature that is a
18 combination of M_1 and the one or more codes (e.g., $(M_1,r,s,auth)$).

19 At 160, it reports the one or more codes and/or the digital signature (e.g.,
20 $(M_1,r,s,auth)$).

21
22 Example:

23 Message $M = \langle M_1, M_2 \rangle$

24 Length of digital signature (e.g., a PID) = 114 bits
25

1 Format of the signature= $(M_1, r, s, auth)$, where M_1 has 11 bits which
2 may denote the site code and upgrade flag, r has 32 bits, s has 64 bits,
3 and $auth$ has 7 bits for an authenticity check.

4 Underlying group for Discrete Log problem: Work in elliptic curve
5 subgroup G of order $q \approx 2^{64}$ modulo a 512-bit prime. This is a larger
6 subgroup and larger modulus than the earlier conventional approaches. This
7 is possible because one may embed several bits (e.g., 20 bits) of the
8 message into the digital signature implicitly.

9 Private key: Exponent x , group order q

10 Semiprivate (known only to the signer and authentication server):

11 Backdoor key BK

12 Public: Generator g for subgroup, $y = g^x$.

13
14 Of course, one may choose another modulus size, such as 1024 bits.

15 But the above subgroup's order q is much smaller: the best known ways to
16 compute discrete logs use a Pollard rho type of algorithm, which takes roughly the
17 same amount of time as solving discrete log in this black box subgroup. In this
18 case the assumption is similar to the subgroup assumption in NIST's (National
19 Institute Of Standards and Technology) DSA (Digital Signature Algorithm) based
20 on discrete log modulo a large prime.

21 Also, if an implementation used special curves, one has to evaluate the
22 effects of attacks that use their special properties.

23 The exemplary short digital signature generator may use a specific curve
24 for which the complex multiplication field is known directly. In this, one may use
25

1 a twist of the curve to obtain another curve whose equation will appear to have
2 different structure.

3 4 Amortized Signature Generator

5 The acts of blocks 120-140 of Fig. 1 may need to be repeated many times to
6 find a value of k that satisfies the conditions. On average, it may need to be
7 repeated $2^{\text{Length}(M_2)}$ times. If one needs to generate thousands or millions of such
8 ciphertext, then this can be quite expensive.

9 When generating a large number of signatures having (e.g., several
10 thousand or millions) having the same M_1 , one may amortize this cost so that it
11 averages only a few (e.g., 14) trials to find k rather than $2^{\text{Length}(M_2)}$ times.

12 The concept is well-known to those skilled in the art as the “coupon
13 collector problem”. In brief, the problem is stated like this: If one randomly
14 throws N balls into n bins, how large must N be so the probability that all bins
15 are nonempty is very close to one? According to the commonly held solution to
16 the problem, it can be shown that N is of the order $n \ln n$.

17 With the exemplary short digital signature generator, if n represents the
18 number of digital signatures that one wants and N represents the number of
19 candidates g^k (with appropriate hashing) one must try, then the estimated work
20 load N/n goes up by a factor of $\ln n$. If n is a million then this factor is around 14.

21 The calculations in the coupon collector problem show that if we do not
22 care if a few bins remain empty (or equivalently if we are less confident that all
23 bins are occupied) then the factor can be lowered.

Exemplary Ciphertext Validation

The exemplary short digital signature validator validates a digital signature $(M_1, r, s, auth)$, at least in part, by determining whether (r, s) actually implicitly contains or hides M_2 .

In an implementation using PID activation for software, this validation may occur at a target client computer and/or at a separate central validation computer that communicatively connects to the target computer.

The following pseudo-code for a function for validating a digital signature (in accordance with the exemplary short digital signature validator) of a message M (which is divided into M_1 and M_2) when given a digital signature $(M_1, r, s, auth)$, where (r, s) implies M_2 . By way of illustration and not limitation, this exemplary client-side implementation does not reference *auth*, although server-side implementations can reference *auth*.

CLIENT_VALIDATE($M_1, r, s, auth$)

$$gk = (g^s \cdot y^{H_2(M_1, r)})^{r+1}$$

$$M_2 = H_0(M_1, gk)$$

Test whether $H_1(M_1, gk) = r$.

If successful, return the pair $M = \langle M_1, M_2 \rangle$.

Fig. 2 shows a methodological implementation of the exemplary short digital signature validator. This methodological implementation may be performed in software, hardware, or a combination thereof.

1 At 210 of Fig. 2, the exemplary short digital signature validator obtains a
2 digital signature (*DS*) having at least three portions, M_1 , r , and s . It may also have
3 more portions, such an authentication tag (*auth*). One example of how it may
4 obtain the digital signature (*DS*) is when a human computer user manually enters it
5 at a client computer. This (*DS*) may be a PID associated with a software product
6 and printed on its packaging.

7 At 220, using a first predefined mathematical function employing M_1 , r ,
8 and/or s , the exemplary short digital signature validator calculates the value of gk .

9 At 230, it determines whether a second predefined mathematical function
10 employing M_1 and gk produces a value equivalent to r .

11 If the value of the product of the second predefined mathematical function
12 employing M_1 and gk is equivalent to r , then proceed to block 260; otherwise, the
13 result is invalid and the process ends.

14 At 240, using a third predefined mathematical function employing M_1 and
15 gk , the exemplary short digital signature validator calculates the value of M_2 .

16 These mathematical functions may include hashing so that specific length
17 results are produced.

18 At 250, it produces a message comprising M_1 and M_2 .

19 At 260, it may report the message and indicate the result of the
20 determination of block 230.

21 Server-Side Implementation

22 The authenticity tag adds another level of security to the digital signature.
23 The relationship between authenticity tag and a pair $\langle M_1, M_2 \rangle$ is established by a
24 random (or pseudorandom) function which is private to the signer. This uses the
25

1 backdoor key (BK). This BK is known only by the original signer and by the
2 centralized (and presumably secure) validation system.

3 In the event (which is more likely the lower L is) of a public key being
4 compromised, the probability that a signature generated by a pirate will get
5 through a server is $2^{-\ell}$ where ℓ is the length of the authenticity tag attached to the
6 signature – this signature can be verified at a central server.

7 The following pseudo-code for a function for validating a digital signature
8 (in accordance with the exemplary short digital signature validator) of a message
9 M (which is divided into M_1 and M_2) when given a digital signature ($M_1, r, s, auth$),
10 where (r, s) implies M_2 .

11 In addition, the following pseudo-code tests whether the given signature is
12 valid and authentic (i.e., could not have been produced by someone who has
13 broken only the public key). This is performed by the centralized validation
14 system.

15
16 SERVER_AUTHENTICATE($M_1, r, s, auth$)

17 Do a client test, namely call CLIENT_AUTHENTICATE($M_1, r, s, auth$)

18 Also check whether $auth$ is correct (using BK).
19

20 Fig. 3 shows another methodological implementation of the exemplary
21 short digital signature validator. This methodological implementation may be
22 performed in software, hardware, or a combination thereof.

23 At 310, the exemplary short digital signature validator performs the
24 functions of Fig. 2. It may obtain the digital signature (DS) via remote
25 communications mechanisms. Examples of such include direct point-to-point

1 telephonic connection, Internet, Local Area Network (LAN), Wide Area Network
2 (WAN), Intranet, and wireless communications. These communications may be
3 accomplished using human intermediaries by the human users calling in to a
4 central call-center or communicating via fax or postal mail.

5 If the above returns a message comprising M_1 and M_2 (which indicates that
6 the "client" validation was successful), then the exemplary short digital signature
7 validator performs, at 320, an additional validation testing using a secret key (BK)
8 that is only known to the signer and the centralized validation system.

9 At 330, it reports the results of such validation.

10 **A Quadratic Approach**

11
12 Alternatively, the mathematical functions and calculations may be non-
13 linear (quadratic, for example) in s . In this case the signature is computed by
14 solving the equations below for the values of r and s :

$$15 \quad r = H_1(M_1, g^k) \quad \text{where} \quad k = s^2 + x H_2(M_1, r)s \mod q$$

16
17
18 In this case the verification is done by computing a candidate gk for g^k
19 using $gk = [g^s y^{H_2(M_1, r)}]^s$ and then testing whether $H_1(M_1, gk) = r$. The signer picks
20 some k , computes r as above, computes $h := H_2(M_1, r)$, and solves a quadratic
21 equation for s modulo the prime q .

22 By suitably choosing the modulus, one may make the square root extraction
23 straight forward and use the standard formula for solution of quadratics, namely

$$24 \quad s = 2^{-1}(-xh \pm \sqrt{x^2 h^2 + 4k}).$$

1 If the required square root does not exist, then the signer can look for
2 another k , which will give a fresh quadratic. Another signer strategy sets $h := h + 1$
3 or $h := H_2(M_{site}, r + 1)$ and tries again until a root is found. The number of
4 iterations may be bounded. The validator tries successive values of h until the
5 signature verifies. If $h := H_2(M_1, r + 1)$ is used, then hash values are random, so the
6 probability that it takes exactly a iterations is 2^{-a} (for $a > 0$) and thus the expected
7 number of trials is 2.

8 The significance of the non-linear approach is that for small values of
9 parameters, it prevents the attacker from optimizing her cryptanalysis search by
10 using some sequential search for the candidate values. This would be possible in
11 case the verification equations compute candidate values of g^k using one of the
12 following:

$$gk = [g^s y^{H_2(M_1, r)}]^{r+1}$$

$$gk = [g^s y^{H_2(M_1)}]^{r+1}$$

16 In these cases an attacker could fix the values of r , M_1 and try successive
17 values of s .

19 Exemplary Computing System and Environment

20 **Fig. 4** illustrates an example of a suitable computing environment 400
21 within which an exemplary short digital signature generator/validator, as described
22 herein, may be implemented (either fully or partially). The computing
23 environment 400 may be utilized in the computer and network architectures
24 described herein.
25

1 The exemplary computing environment 400 is only one example of a
2 computing environment and is not intended to suggest any limitation as to the
3 scope of use or functionality of the computer and network architectures. Neither
4 should the computing environment 400 be interpreted as having any dependency
5 or requirement relating to any one or combination of components illustrated in the
6 exemplary computing environment 400.

7 The exemplary short digital signature generator/validator may be
8 implemented with numerous other general purpose or special purpose computing
9 system environments or configurations. Examples of well-known computing
10 systems, environments, and/or configurations that may be suitable for use include,
11 but are not limited to, personal computers, server computers, thin clients, thick
12 clients, hand-held or laptop devices, multiprocessor systems, microprocessor-
13 based systems, set top boxes, programmable consumer electronics, network PCs,
14 minicomputers, mainframe computers, distributed computing environments that
15 include any of the above systems or devices, and the like.

16 The exemplary short digital signature generator/validator may be described
17 in the general context of computer-executable instructions, such as program
18 modules, being executed by a computer. Generally, program modules include
19 routines, programs, objects, components, data structures, etc. that perform
20 particular tasks or implement particular abstract data types. The exemplary short
21 digital signature generator/validator may also be practiced in distributed
22 computing environments where tasks are performed by remote processing devices
23 that are linked through a communications network. In a distributed computing
24 environment, program modules may be located in both local and remote computer
25 storage media including memory storage devices.

1 The computing environment 400 includes a general-purpose computing
2 device in the form of a computer 402. The components of computer 402 may
3 include, by are not limited to, one or more processors or processing units 404, a
4 system memory 406, and a system bus 408 that couples various system
5 components including the processor 404 to the system memory 406.

6 The system bus 408 represents one or more of any of several types of bus
7 structures, including a memory bus or memory controller, a peripheral bus, an
8 accelerated graphics port, and a processor or local bus using any of a variety of
9 bus architectures. By way of example, such architectures may include an Industry
10 Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an
11 Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA)
12 local bus, and a Peripheral Component Interconnects (PCI) bus also known as a
13 Mezzanine bus.

14 Computer 402 typically includes a variety of computer readable media.
15 Such media may be any available media that is accessible by computer 402 and
16 includes both volatile and non-volatile media, removable and non-removable
17 media.

18 The system memory 406 includes computer readable media in the form of
19 volatile memory, such as random access memory (RAM) 410, and/or non-volatile
20 memory, such as read only memory (ROM) 412. A basic input/output system
21 (BIOS) 414, containing the basic routines that help to transfer information
22 between elements within computer 402, such as during start-up, is stored in ROM
23 412. RAM 410 typically contains data and/or program modules that are
24 immediately accessible to and/or presently operated on by the processing unit 404.
25

1 Computer 402 may also include other removable/non-removable,
2 volatile/non-volatile computer storage media. By way of example, Fig. 4
3 illustrates a hard disk drive 416 for reading from and writing to a non-removable,
4 non-volatile magnetic media (not shown), a magnetic disk drive 418 for reading
5 from and writing to a removable, non-volatile magnetic disk 420 (e.g., a "floppy
6 disk"), and an optical disk drive 422 for reading from and/or writing to a
7 removable, non-volatile optical disk 424 such as a CD-ROM, DVD-ROM, or other
8 optical media. The hard disk drive 416, magnetic disk drive 418, and optical disk
9 drive 422 are each connected to the system bus 408 by one or more data media
10 interfaces 426. Alternatively, the hard disk drive 416, magnetic disk drive 418,
11 and optical disk drive 422 may be connected to the system bus 408 by one or more
12 interfaces (not shown).

13 The disk drives and their associated computer-readable media provide non-
14 volatile storage of computer readable instructions, data structures, program
15 modules, and other data for computer 402. Although the example illustrates a hard
16 disk 416, a removable magnetic disk 420, and a removable optical disk 424, it is to
17 be appreciated that other types of computer readable media which may store data
18 that is accessible by a computer, such as magnetic cassettes or other magnetic
19 storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or
20 other optical storage, random access memories (RAM), read only memories
21 (ROM), electrically erasable programmable read-only memory (EEPROM), and
22 the like, may also be utilized to implement the exemplary computing system and
23 environment.

24 Any number of program modules may be stored on the hard disk 416,
25 magnetic disk 420, optical disk 424, ROM 412, and/or RAM 410, including by

1 way of example, an operating system 426, one or more application programs 428,
2 other program modules 430, and program data 432.

3 A user may enter commands and information into computer 402 via input
4 devices such as a keyboard 434 and a pointing device 436 (e.g., a "mouse").
5 Other input devices 438 (not shown specifically) may include a microphone,
6 joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and
7 other input devices are connected to the processing unit 404 via input/output
8 interfaces 440 that are coupled to the system bus 408, but may be connected by
9 other interface and bus structures, such as a parallel port, game port, or a universal
10 serial bus (USB).

11 A monitor 442 or other type of display device may also be connected to the
12 system bus 408 via an interface, such as a video adapter 444. In addition to the
13 monitor 442, other output peripheral devices may include components such as
14 speakers (not shown) and a printer 446 which may be connected to computer 402
15 via the input/output interfaces 440.

16 Computer 402 may operate in a networked environment using logical
17 connections to one or more remote computers, such as a remote computing device
18 448. By way of example, the remote computing device 448 may be a personal
19 computer, portable computer, a server, a router, a network computer, a peer device
20 or other common network node, and the like. The remote computing device 448 is
21 illustrated as a portable computer that may include many or all of the elements and
22 features described herein relative to computer 402.

23 Logical connections between computer 402 and the remote computer 448
24 are depicted as a local area network (LAN) 450 and a general wide area network
25

1 (WAN) 452. Such networking environments are commonplace in offices,
2 enterprise-wide computer networks, intranets, and the Internet.

3 When implemented in a LAN networking environment, the computer 402 is
4 connected to a local network 450 via a network interface or adapter 454. When
5 implemented in a WAN networking environment, the computer 402 typically
6 includes a modem 456 or other means for establishing communications over the
7 wide network 452. The modem 456, which may be internal or external to
8 computer 402, may be connected to the system bus 408 via the input/output
9 interfaces 440 or other appropriate mechanisms. It is to be appreciated that the
10 illustrated network connections are exemplary and that other means of establishing
11 communication link(s) between the computers 402 and 448 may be employed.

12 In a networked environment, such as that illustrated with computing
13 environment 400, program modules depicted relative to the computer 402, or
14 portions thereof, may be stored in a remote memory storage device. By way of
15 example, remote application programs 458 reside on a memory device of remote
16 computer 448. For purposes of illustration, application programs and other
17 executable program components such as the operating system are illustrated herein
18 as discrete blocks, although it is recognized that such programs and components
19 reside at various times in different storage components of the computing device
20 402, and are executed by the data processor(s) of the computer.

21 22 Computer-Executable Instructions

23 An implementation of an exemplary short digital signature
24 generator/validator may be described in the general context of computer-
25 executable instructions, such as program modules, executed by one or more

1 computers or other devices. Generally, program modules include routines,
2 programs, objects, components, data structures, etc. that perform particular tasks
3 or implement particular abstract data types. Typically, the functionality of the
4 program modules may be combined or distributed as desired in various
5 embodiments.

6 7 Exemplary Operating Environment

8 Fig. 4 illustrates an example of a suitable operating environment 400 in
9 which an exemplary short digital signature generator/validator may be
10 implemented. Specifically, the exemplary short digital signature
11 generator/validator(s) described herein may be implemented (wholly or in part) by
12 any program modules 428-430 and/or operating system 426 in Fig. 4 or a portion
13 thereof.

14 The operating environment is only an example of a suitable operating
15 environment and is not intended to suggest any limitation as to the scope or use of
16 functionality of the exemplary short digital signature generator/validator(s)
17 described herein. Other well-known computing systems, environments, and/or
18 configurations that are suitable for use include, but are not limited to, personal
19 computers (PCs), server computers, hand-held or laptop devices, multiprocessor
20 systems, microprocessor-based systems, programmable consumer electronics,
21 wireless phones and equipments, general- and special-purpose appliances,
22 application-specific integrated circuits (ASICs), network PCs, minicomputers,
23 mainframe computers, distributed computing environments that include any of the
24 above systems or devices, and the like.

Computer Readable Media

An implementation of an exemplary short digital signature generator/validator may be stored on or transmitted across some form of computer readable media. Computer readable media may be any available media that may be accessed by a computer. By way of example, and not limitation, computer readable media may comprise “computer storage media” and “communications media.”

“Computer storage media” include volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules, or other data. Computer storage media include, but are not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which may be used to store the desired information and which may be accessed by a computer.

“Communication media” typically embody computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as carrier wave or other transport mechanism. Communication media also include any information delivery media.

The term “modulated data signal” means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media include wired media such as a wired network or direct-wired connection, and wireless

1 media such as acoustic, RF, infrared, and other wireless media. Combinations of
2 any of the above are also included within the scope of computer readable media.

3 **Conclusion**

4
5 Although the invention has been described in language specific to structural
6 features and/or methodological steps, it is to be understood that the invention
7 defined in the appended claims is not necessarily limited to the specific features or
8 steps described. Rather, the specific features and steps are disclosed as preferred
9 forms of implementing the claimed invention.
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25